MANUALES DE USUARIO Y DE BACKEND

APP CHECKEASY

Autores:

Juan Diego Lasso Montaño

Juan José Plazas Vargas

Valery Karina Martínez Vanegas

Laura Natalia Duarte Acero

Docente: Néstor German Bolivar Pulgarin

Programación Orientada a Objetos -2016375

2024 I GRUPO

ÍNDICE

ÍNDICE DE FIGURAS

Introducción

El aplicativo CHECKEASY, es una aplicación gráfica que permite realizar las principales funciones que realiza un hotel. Como control de empleados, habitaciones, reservas, check in, check out, consultas y cancelaciones.

El manual de usuario de CHECKEASY, muestra la funcionalidad del programa, sus vistas y opciones para la interacción con el usuario.

El manual de backend de CHECKEASY, corresponde a una guía integral para una comprensión del desarrollo en el lado del servidor, gestionando la parte lógica, bases de datos y la interacción con otros servicios, dando a conocer los requerimientos para su funcionamiento de hardware, software y base de datos, donde por secciones cada uno de los diagramas de secuencia como son: Reserva/Checkin, Checkout, Cancelar reserva y consulta.

MANUAL DEL USUARIO

El manual del usuario de la aplicación CheckEasy se encuentra en el siguiente enlace:

https://youtu.be/nUouDgwj8yE?si=y5pi0l6m1FY3l7vy

MANUAL DE BACKEND

1. Arquitectura Tecnológica

1.1. Modelo Operativo

A nivel hardware y software el aplicativo requiere:

1. Windows 10
2. MySQL 8.036
3. A través de Apache Netbeans IDE 20 como entorno de desarrollo.
4. Java y Java(TM)SE Development Kit 17.0.10

1.2. Gestión de Datos

En MySQL se crea el schema ***bdhotel***. y las siguientes tablas con sus respectivos campos y tipo de dato.

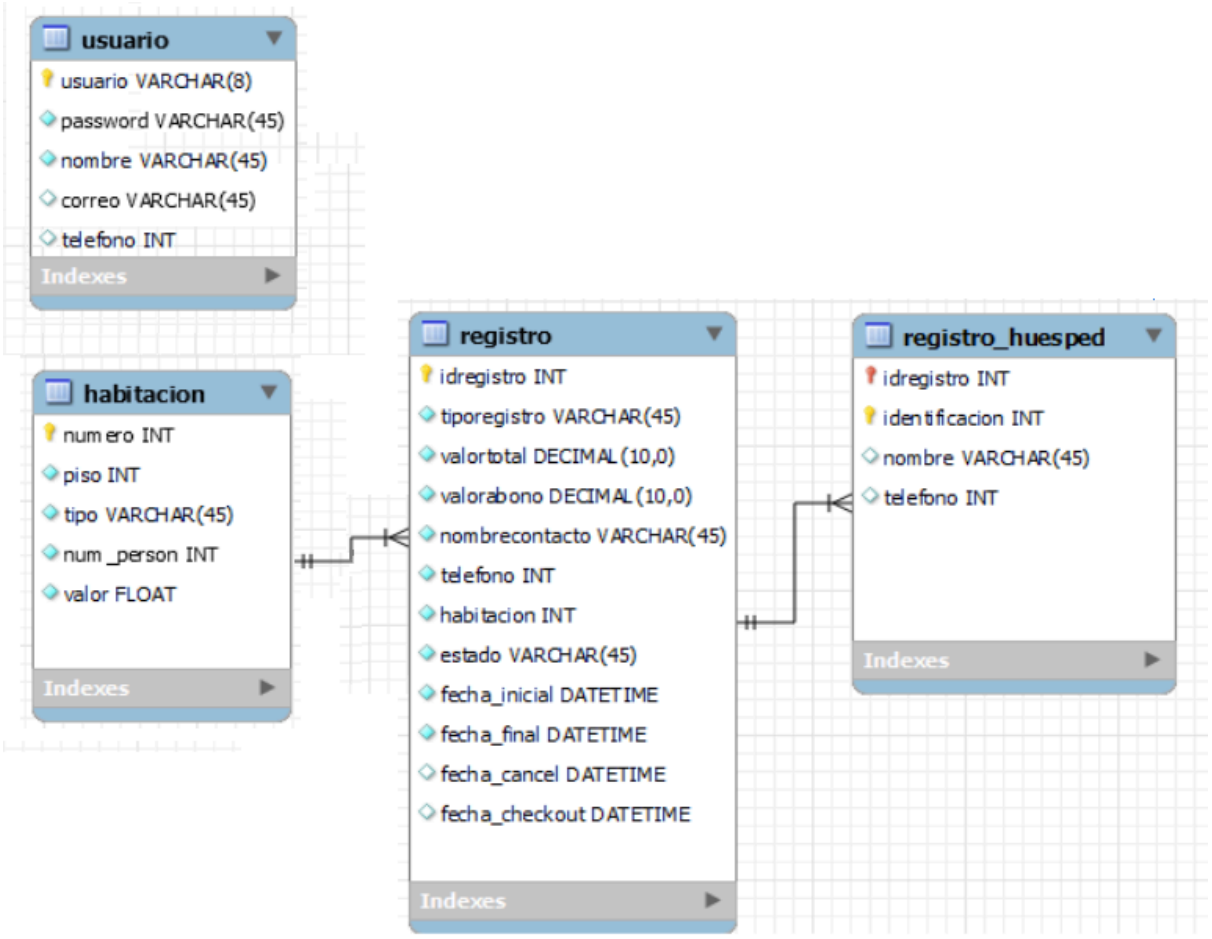

Figura 1.1: Diagrama Entidad Relación

En la tabla usuario se almacena los empleados que utilizan el aplicativo.

En la tabla habitación se almacenan los datos principales de una habitación como número y piso.

En la tabla registro se almacena la información requerida para realizar el registro de check in o reserva, como habitación, nombre del contacto, fechas y valor tanto de registro como de abono.

En la tabla registro huésped se almacena la información de las personas que se van a alojar.

1.3. Organización del aplicativo

Los archivos del proyecto se agruparon en paquetes funcionales:

- ★ **Model**: se almacenarán las clases.
- ★ **View**: Se almacenarán los formularios o interfaces gráficas para cada proceso.
- ★ **Images** : Se almacenarán las imágenes que se utilizaran
- ★ **Controller** : Se almacenará la conexión con la base de datos.

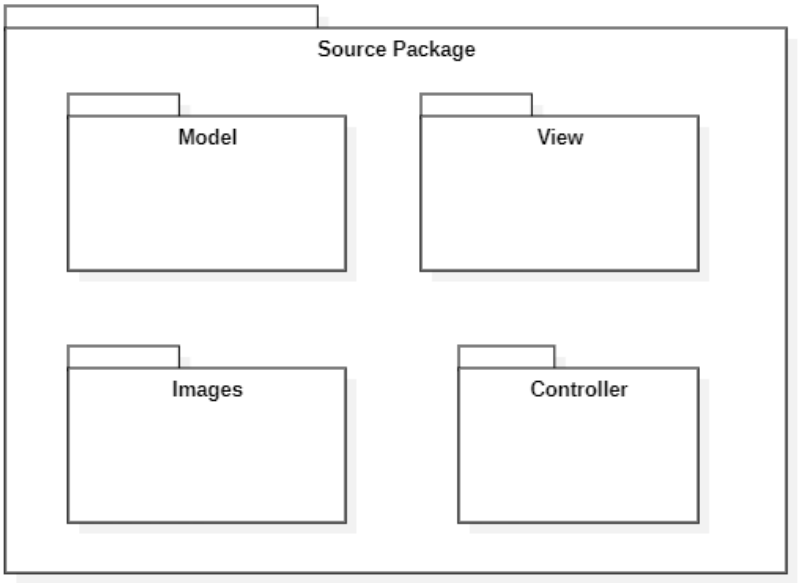

Figura 1.2: Diagrama de Paquetes

1.4. Diagrama de Clases

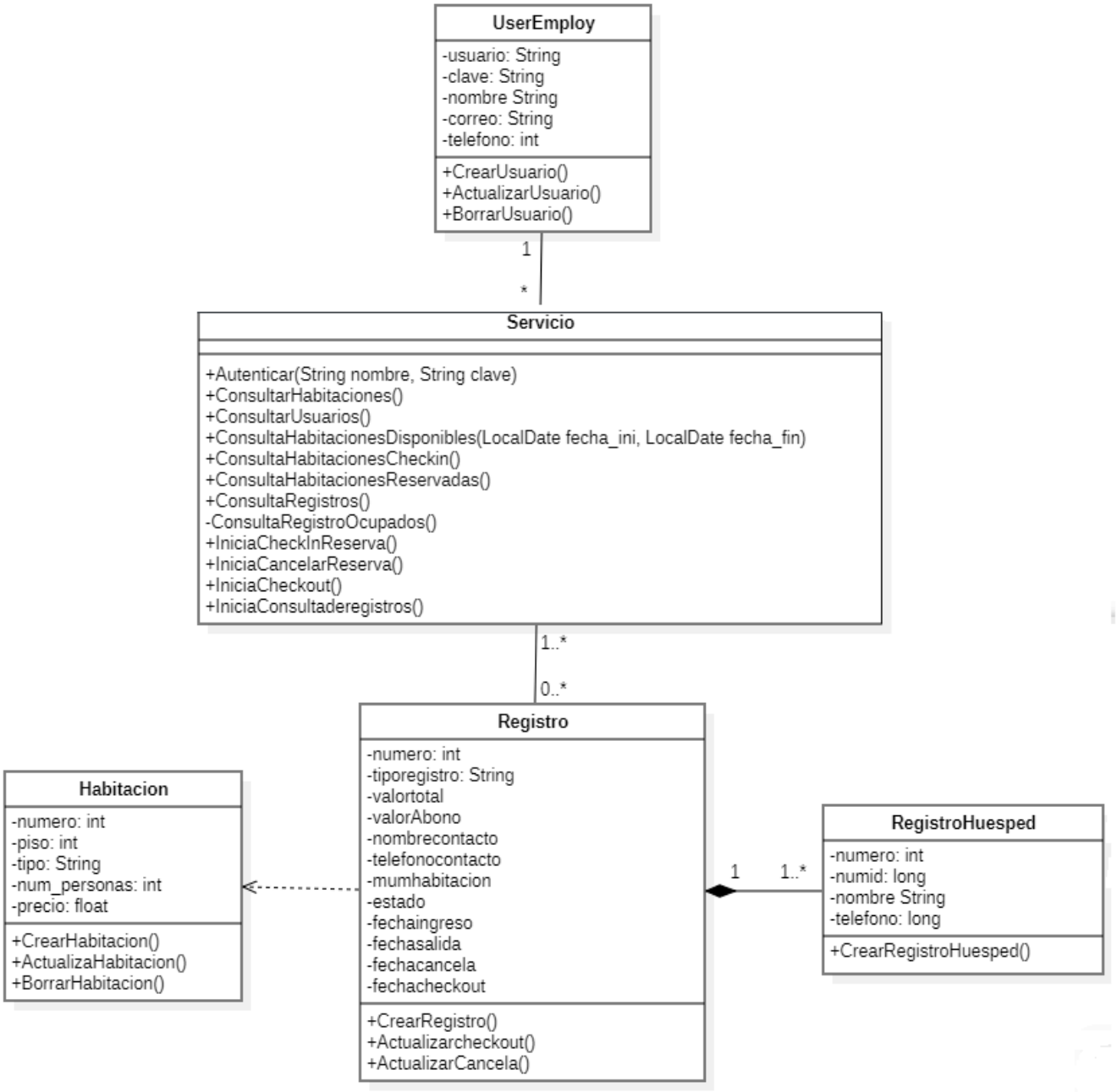

Figura 1.3: Diagrama de Clases

En la figura 2.3, se pueden ver las diferentes clases relacionadas entre sí. Cada una se detalla a continuación:

UserEmploy: Mantiene la información personal de los usuarios, es decir la persona que está haciendo uso de la aplicación, que sería un empleado del hotel. Sus atributos mayormente son de tipo String como son el usuario, la contraseña, el nombre completo y el e-mail, y lo que no lo son valores Integer como el teléfono.
Mediante sus métodos puede instanciar usuarios, así como borrarlos y modificar la información que tienen en sus atributos.

RegistroHuesped: Almacena la información de los diferentes huéspedes que pasan por el hotel, como es el nombre, un documento identificativo y el teléfono de contacto. Así como almacena el número del registro con el cual se quedaron en el hotel.Por lo que tiene una relación de composición con la clase Registro.

Registro: Almacena la información que debe tener cualquiera de las acciones que necesita realizar un hotel, ya sea la informacion de la habitacion que el huesped haya solicitado, como la información de contacto del húesped y las fechas de estadia, además de tener opción de almacenar la fecha cuando se realice el checkout y un campo para almacenar la fecha en caso de Cancelar una reserva. A través de sus metodos podemos crear un registro y actualizar un registro en caso de realizar el checkout o la cancelación de una reserva.

Habitación: Mantiene la información de cada una de las habitaciones del hotel, sus atributos son el número de la habitacion, el piso en el que se encuentra ubicada, su tipo(SENCILLA, DOBLE…), el número de personas que pueden disponer de la habitacion y el precio por noche.Mediante sus métodos puede instanciar habitacioness, así como borrarlas y modificar la información que tienen en sus atributos.

Servicio: la clase servicio es donde están los métodos principales para las acciones CHECKIN/RESERVA, CHECKOUT, CONSULTA Y CANCELAR RESERVA, algunos de sus métodos son:

1. **Autenticar():** que mediante un usuario y una contraseña determina que dicho usuario tenga acceso a la aplicación.
2. **Consultarhabitaciones():**que retorna la información de todas las habitaciones.
3. **ConsultarUsuarios():** que retorna la información de todos los usuarios .
4. **ConsultarHabitacionesDisponibles(LocalDate fecha_ini, LocalDate fecha_fin):** que retorna las habitaciones que se encuentren libres en el periodo de tiempo entre fecha_ini y fecha_fin.
5. **ConsultaHabitacionesCheckin():** que retorna la información de las habitaciones que están en los registros cuyo estado sea “CHECKIN”.
6. **ConsultaHabitacionesReservadas():** que retorna la información de las habitaciones que están en los registros cuyo estado sea “RESERVADO”.
7. ConsultaRegistros(): que retorna la información de todos los registros.
8. **ConsultaRegistrosOcupados():**que retorna la información de las habitaciones que están en los registros cuyo estado sea “OCUPADO”.
9. **IniciaCheckinReserva():** se muestra la ventana de “Checkin/Reserva" en la pantalla.
10. **IniciaCancelarReserva():se muestra la ventana de** “Checkin/Reserva" en la pantalla.
11. **IniciaCheckout():**se muestra la ventana de “Checkout" en la pantalla.
12. **IniciaConsultaderegistros():**se muestra la ventana de “Consulta" en la pantalla.

2. Sección 1. Checkin/Reserva

2.1. Diagrama de Secuencia

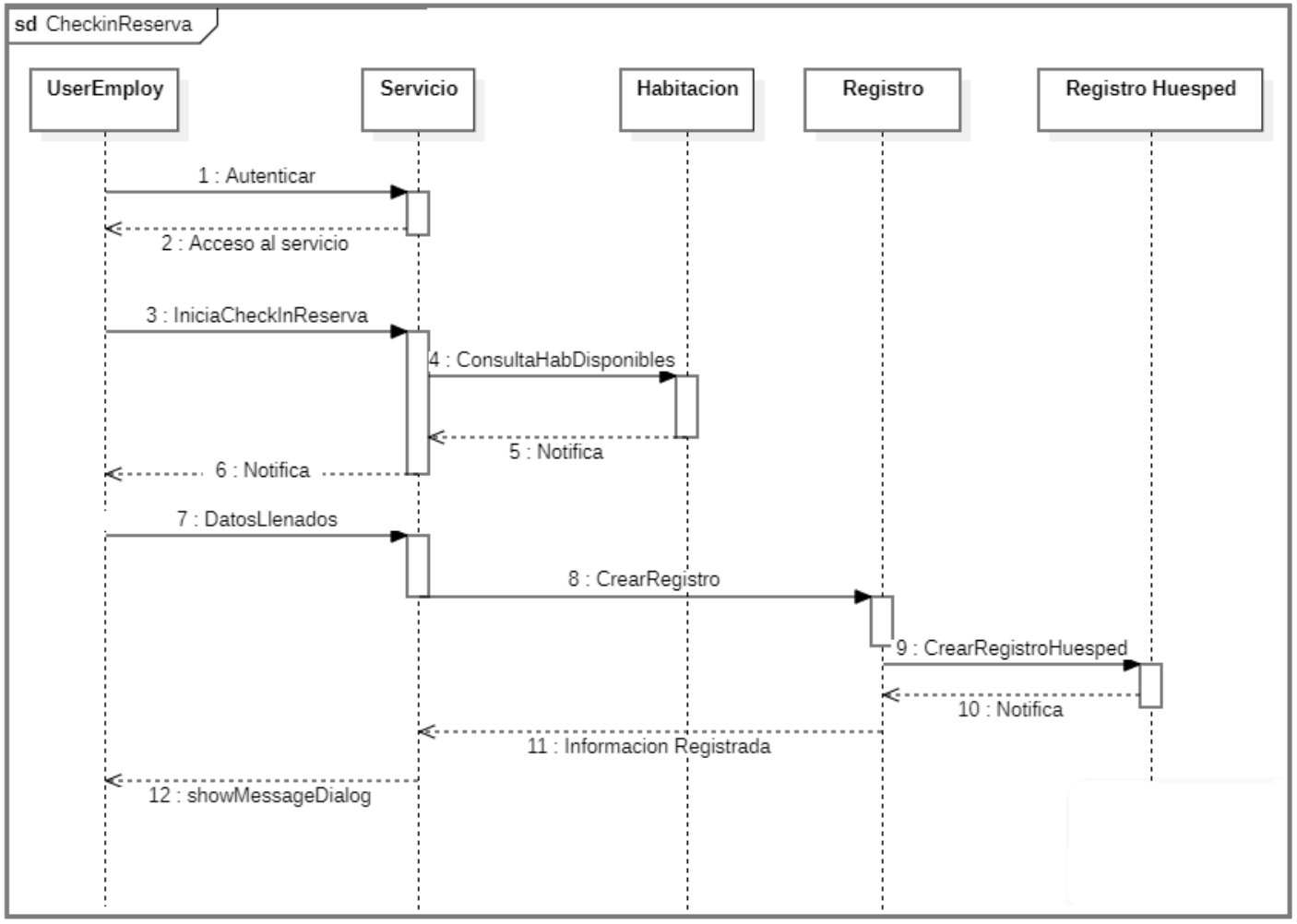

Figura 2.1: Diagrama de Secuencia CheckinReserva

2.2. Descripción de acciones:

1. **Autenticar():** Es un método que se encuentra en la clase **Servicio**, el cual con el usuario y clave que ingresa el empleado, verifica que el usuario exista en la tabla "usuario" de la base de datos y que la clave esté correcta, y así autentica para ingresar

al aplicativo y darle acceso al servicio al usuario/empleado que esté haciendo uso de la aplicación.

2. **Acceso al servicio():** La clase **Servicio** notifica el acceso o no al aplicativo mediante un JOptionPane.

3. **InciaCheckinReserva()**: Es un método de la clase **Servicio**, el cual mediante la selección de la opción Check In/Reserva en el menú, abre el panel Check In/Reserva para que la persona pueda hacer uso del servicio, así como le pone un nombre en la parte de arriba de la ventana.

4. **ConsultaHabDisponibles()**: Método de la clase **Servicio,** después de que el usuario haya puesto las fechas en que se van a quedar los huéspedes, la acción que realiza es ir a la Base de Datos e identificar los registros para esas fechas y que número de habitación tienen para posteriormente enviar esos datos a la clase Habitacion para identificar cuales son las habitaciones disponibles en una fechas determinadas (Ingreso y salida),

5. **Notifica**: La clase **Habitación** ,mediante una búsqueda en la tabla "habitación", notifica a través de una lista de tipo Habitación con las habitaciones disponibles en esas fechas a la clase **Servicio**.

6. **Notifica**: La clase **Servicio** notifica a la pantalla los datos enviados de habitaciones disponibles con los valores de las habitaciones mediante una tabla con datos de las habitaciones disponibles.

7. **DatosLlenados:** El usuario selecciona qué habitacion quieren los huéspedes, llena la información sobre el/los huésped(s) que va(n) a hacer el check in o reserva, además de los valores de abono.

8. **CrearRegistro:** Luego de validar que todos los datos estén completos, la clase **Registro** crea un registro en la base de datos con la información digitada, con el contacto, así como de la habitación que se va a usar, con estado "OCUPADO" o "RESERVADO" respectivamente.

9. **CrearRegistroHuesped:** La clase **RegistroHuesped**, crea uno a más registros en la base de datos, tabla "registro_huesped" con la información de los huéspedes, cada registro con el id del registro del contacto que quedó en la tabla "registro".

10. **Notifica:** La clase **RegistroHuesped** notifica a **Registro** que fue correcto el ingreso.

11. I**nformación Registrada**: La clase **Registro** le dice a **Servicio** que fue correcto el ingreso del registro a la base de datos.

12. **showMessageDialog**: La clase **Servicio** notifica al usuario que el registro fue exitoso mediante un JOptionPane.showMessageDialog que dice "Proceso realizado con éxito".

3. Sección 2. Checkout

3.1. Diagrama de Secuencia

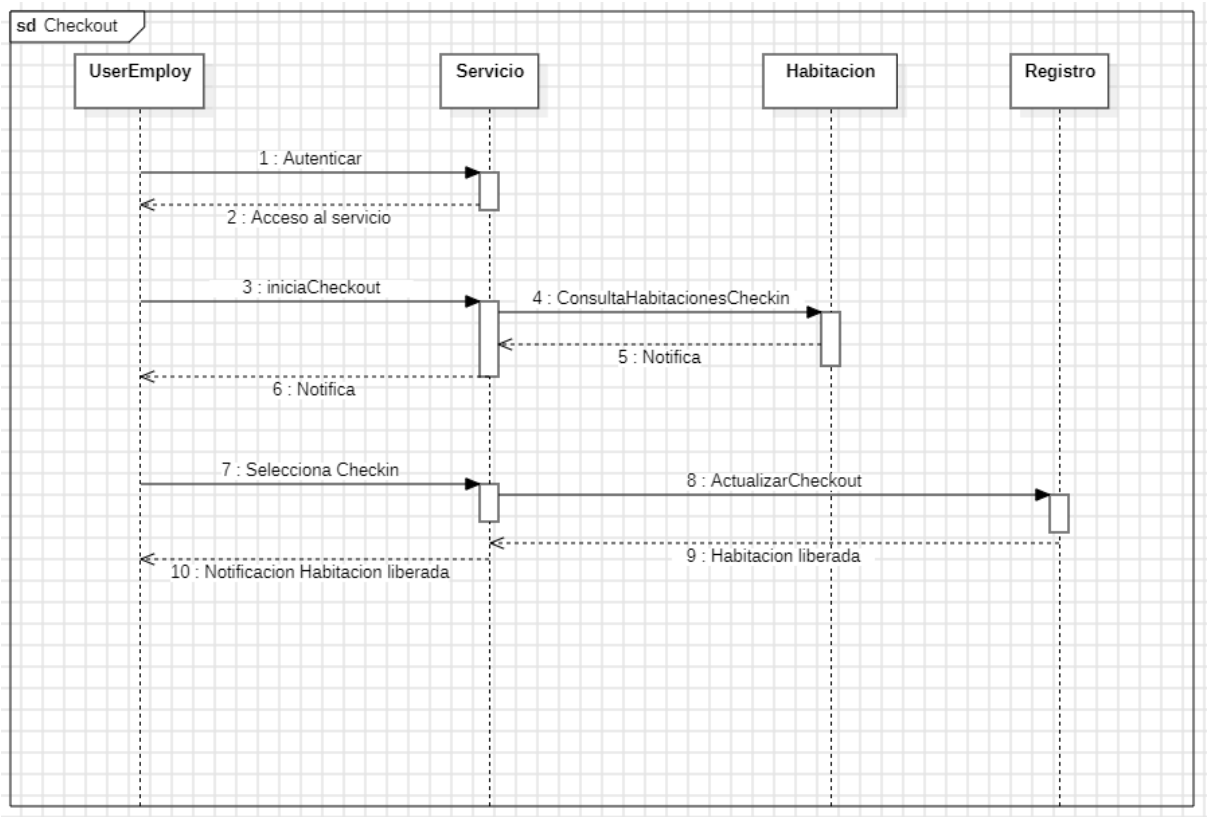

Figura 3.1: Diagrama de Secuencia Checkout

3.2. Descripción de acciones:

1. **Autenticar():** Es un método que se encuentra en la clase **Servicio**, el cual con el usuario y clave que ingresa el empleado, verifica que el usuario exista en la tabla "usuario" de la base de datos y que la clave esté correcta, y así autentica para ingresar al aplicativo y darle acceso al servicio al usuario/empleado que esté haciendo uso de la aplicación.

2. **Acceso al servicio():** La clase **Servicio** notifica el acceso o no al aplicativo mediante un JOptionPane.

3. **IniciaCheckOut()**: Es un método de la clase **Servicio**, el cual mediante la selección de la opción CheckOut en el menú, abre el panel Check Out para que la persona pueda hacer uso del servicio, así como le pone un nombre en la parte de arriba de la ventana.

4. ConsultaHabitacionesCheckin(): Método de la clase **Servicio,** al abrir la ventana esta se conecta a la Base de Datos e identifica los registros que tengan estado **"CHECKIN"** para posteriormente enviar esos datos a la clase **Habitacion** que retornará la información de los registros que cumplan esa condición.

5. Notifica: La clase **Habitación**, mediante una búsqueda en la tabla "habitación", notifica a través de una lista de tipo Habitación con las habitaciones en Checkin a la clase **Servicio**.

6. Notifica: La clase **Servicio** notifica a la pantalla los datos enviados mediante una tabla con datos de las habitaciones reservadas(id, numero, valor, contacto, telefono, fecha de llegada, fecha de salida).

7. Selecciona Checkin: El usuario selecciona qué registro pertenece al cliente que quiere cancelar su reserva.

8. ActualizarCheckout(): Luego de validar con el cliente sobre la decisión de realizar el Checkout, la clase **Registro** actualiza el estado del registro de la habitación reservada en la base de datos de **"CHECKIN"** a "**CHECKOUT".**

9. Habitación liberada: La clase **Registro** notifica a la clase **Servicio** que fue correcta la actualización del registro a la base de datos.

10. Notifica: La clase **Servicio** notifica al usuario de que el proceso fue realizado con éxito mediante un JOptionPane.showMessageDialog.

4. Sección 3.Cancelar Reserva

4.1. Diagrama de Secuencia

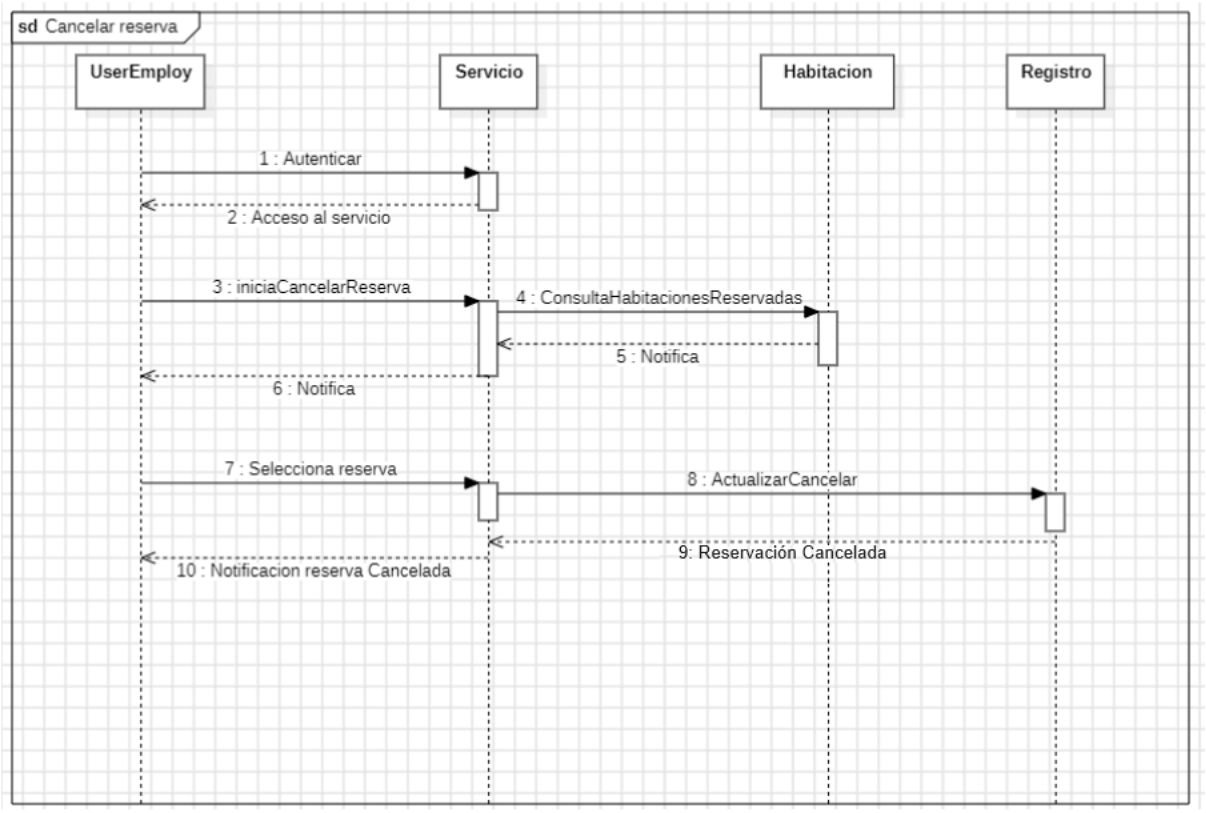

Figura 4.1: Diagrama de Secuencia Cancelar Reserva.

4.2. Descripción de acciones:

1. **Autenticar():** Es un método que se encuentra en la clase **Servicio**, el cual con el usuario y clave que ingresa el empleado, verifica que el usuario exista en la tabla "usuario" de la base de datos y que la clave esté correcta, y así autentica para ingresar al aplicativo y darle acceso al servicio al usuario/empleado que esté haciendo uso de la aplicación.

2. **Acceso al servicio():** La clase **Servicio** notifica el acceso o no al aplicativo mediante un JOptionPane.

3. **InciaCancelarReserva()**: Es un método de la clase **Servicio**, el cual mediante la selección de la opción Cancelar Reserva en el menú, abre el panel Cancelar Reserva para que la persona pueda hacer uso del servicio, así como le pone un nombre en la parte de arriba de la ventana.

4. **ConsultaHabitacionesReservadas()**: Método de la clase **Servicio,** al abrir la ventana esta se conecta a la Base de Datos e identifica los registros que tengan estado **"RESERVADA"** para posteriormente enviar esos datos a la clase **Habitacion** que retornará la información de los registros que cumplan esa condición.

5. **Notifica**: La clase **Habitación**, mediante una búsqueda en la tabla "habitación", notifica a través de una lista de tipo Habitación con las habitaciones reservadas a la clase **Servicio**.

6. **Notifica**: La clase **Servicio** notifica a la pantalla los datos enviados mediante una tabla con datos de las habitaciones reservadas(id, numero, valor, contacto, telefono, fecha de llegada, fecha de salida).

7. **Selecciona reserva:** El usuario selecciona qué registro pertenece al cliente que quiere cancelar su reserva.

8. **ActualizarCancelar():** Luego de validar con el cliente sobre la decisión de Cancelar su reserva, la clase **Registro** actualiza el estado del registro de la habitación reservada en la base de datos de **"RESERVADA"** a "**CANCELADA"**.

9. **Reservación cancelada:**La clase **Registro** notifica a la clase **Servicio** que fue correcta la actualización del registro a la base de datos.

10. **Notifica:** La clase **Servicio** notifica al usuario de que el proceso fue realizado con éxito mediante un JOptionPane.showMessageDialog.

5. Sección 3. Consultas

5.1. Diagrama de Secuencia

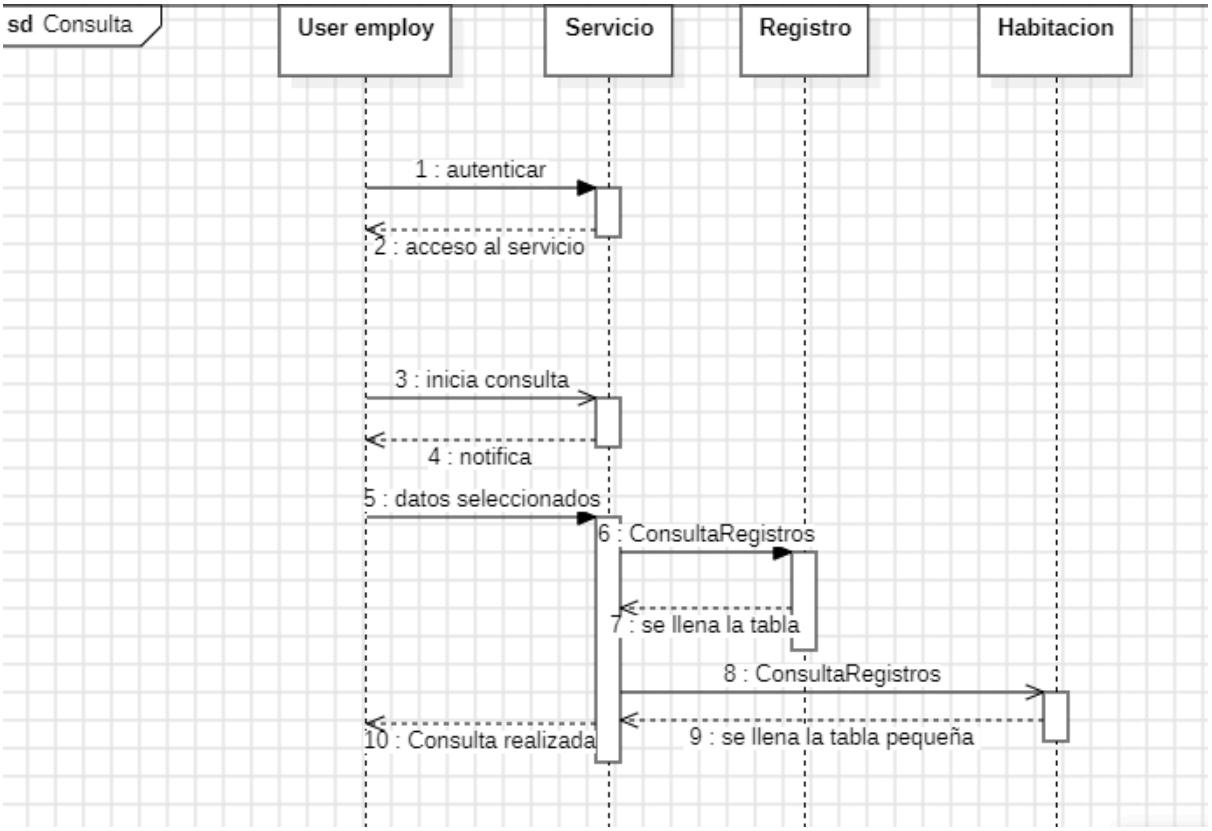

Figura 5.1: Diagrama de Secuencia Consulta.

5.2. Descripción de acciones:

1. **Autenticar():** Es un método que se encuentra en la clase **Servicio**, el cual con el usuario y clave que ingresa el empleado, verifica que el usuario exista en la tabla "usuario" de la base de datos y que la clave esté correcta, y así autentica para ingresar al aplicativo y darle acceso al servicio al usuario/empleado que esté haciendo uso de la aplicación.

2. **Acceso al servicio():** Notifica el acceso a la aplicación o el no acceso mediante un JOptionPane.

3. **IniciaConsulta():** Es un método de la clase **Servicio**, el cual mediante la selección de la opción Consulta en el menú, abre el panel Consulta para que la persona pueda hacer uso del servicio, así como le pone un nombre en la parte de arriba de la ventana.
4. **Notifica():** Notifica al usuario que ya puede empezar el proceso de consulta luego de seleccionarlo en la ventana de procesos.

5. **Datos Seleccionados():** Es un método de la clase Servicio que revisa si se han seleccionado los datos para realizar el proceso de consulta y permitir su ejecución.

6. **ConsultaRegistros():** Es un método de la clase Servicio. Luego de que los datos hayan sido seleccionados y el usuario haya hecho clic en 'consultar', este método accede a la base de datos y guarda los registros según el filtro del usuario.

7. **Se llena la tabla():** Luego de que se llenen los datos y se haga uso del método “ConsultaRegistros()”, se llena la tabla con los registros de interés del usuario.

8. **Se llena la tabla pequeña():** Si se llenan los datos con una habitación en específico y esta habitación tiene dos registros con una brecha entre el fin de uno y el inicio del siguiente, se llenará la tabla con el rango de fechas en el que dicha habitación está libre.

9. **Consulta realizada():** Notifica al usuario que la consulta ha sido realizada con éxito.